|  |  |
| --- | --- |
| Materi | Nilai |
| Pembelajaran CodeIgniter 4 Video 35 – 52  (Tugas minggu ke – 20) | 95 |

**Saya Sudah Belajar dan Mengerti dan Saya BISA**

1. Menampilkan isi dari vorder menggunakan perintah SELECT dan mengirimkan data ke dalam view.

Cth :

public function index()

    {

        $db = \Config\Database::connect();

        $sql = "SELECT \* FROM vorder ORDER BY status ASC";

        $result = $db -> query($sql);

        $row = $result -> getResult('array');

$data = [

            'order' => $row

        ];

*return* view('order/select', $data);

}

Artinya, di dalam function index terdapat $db yang digunakan untuk menyambungkan ke dalam database, $sql berisi perintah SELECT data / memilih data dari vorder yang diurutkan dari status dengan menggunakan urutas ASC yaitu dari kecil ke besar. Lalu di dalam $result terdapat $db yang menjalankan perintah query pada $sql. $row berisi $result yang menjalankan perintah getResult / perintah untuk mendapatkan data dan hasil dari getResult adalah array, jadi hasil yang ditampilkan akan berupa array, apabila tidak diberi ‘array’ di dalam getResult(), maka hasilnya akan berupa objek. Terdapat array assosiatif yaitu $data yang digunakan untuk menyimpan data yang akan dikirimkan ke dalam view. Di dalam $data terdapat key order dengan value yaitu menjalankan $row. Setelah itu, akan menjalankan view order/select dan juga akan menjalankan $data.

1. Membuat tabel untuk menampilkan data yang diambil dengan kolom – kolom sesuai kolom yang ada pada vorder.
2. Melakukan pengujian menggunakan if apabila kolom status bernilai 1 maka akan menampilkan tulisan LUNAS.

Cth :

 <?php

*if* ($value['status'] === 1) {

                  $status = "LUNAS";

              } *else* {

                  $status = "BELOM LUNAS";

              }

                    ?>

                    <td><?= $status?></td>

Artinya, terdapat pengujian apabila $value[‘status’] bernilai sama dengan 1, maka akan menampilkan $status yang berisi tulisan “LUNAS”, tetapi apabila nilai dari $value[‘status’] bernilai tidak sama dengan 1 maka akan menampilkan tulisan “BELOM LUNAS”. Setelah melakukan pengujian, $status dibaca di dalam tabel.

1. Mengatur paging dari halaman yang menampilkan data pelanggan pada order.

Cth :

Pada function index di dalam file order.php

public function index()

    {

        $sql = "SELECT \* FROM vorder";

        $result = $db -> query($sql);

        $row = $result -> getResult('array');

        $total = count($row);

        $tampil = 3;

*if* (isset($\_GET['page'])) {

            $page = $\_GET['page'];

            $mulai = ($tampil \* $page) - $tampil;

            $sql = "SELECT \* FROM vorder ORDER BY status ASC LIMIT $mulai, $tampil";

        }*else*{

            $sql = "SELECT \* FROM vorder ORDER BY status ASC LIMIT 0, $tampil";

        }

        $result = $db -> query($sql);

        $row = $result -> getResult('array');

        $data = [

            'order' => $row,

            'pager' => $pager,

            'perPage' => $tampil,

            'total' => $total

        ];

*return* view('order/select', $data);

    }

Artinya, di dalam function index terdapat $sql yang mengambil data / SELECT data dari vorder, lalu $result berisi $db yang menjalankan perintah query pada $sql, di dalam $row berisi $result yang menjalankan perintah getResult / mendapatkan data yang hasil data yang ditampilkan adalah berupa array. Setelah itu $total menghitung jumlah data yang ada pada $row dengan menggunakan perintah count, $tampil diatur dengan nilai 3. Terdapat pengujian yaitu, apabila $\_GET[‘page’] di klik, maka $page mengambil nilai dari $\_GET[‘page’], lalu $mulai menyimpan hasil operasi aritmatika yang menghitung $tampil dikalikan $page lalu dikurangi dengan $tampil. Contoh, $page adalah 3, maka $tampil \* $page - $tampil, jadi 3 \* 3 – 3 = 6, jadi pada apabila $page bernilai 3, maka nilai yang ditampilkan adalah dimulai dari 7. Setelah mendapatkan nilai daro $mulai, lalu $sql berisi menjalankan perintah untuk memilih data / SELECT data dari vorder yang diurutkan dari status dengan urutan ASC atau mengurutkan dari yang kecil ke yang besa dengan menampilkan $mulai, data dimulai dari nomor berapa dan $tampil, data yang ditampilkan adalah berapa banyak. Dan apabila $\_GET[‘page’] tidak di klik, maka akan menjalankan $sql yang berisi memilih data / SELECT data dari vorder yang diurutkan dari status dengan urutan ASC atau mengurutkan data yang brnilai kecil ke besar dengan data yang ditampilkan dimulai dari 0 dan data yang ditampilkan sebanyak $tampil / 3. Setelah itu, $result berisi $db yang menjalankan perintah query pada $sql, lalu $row berisi $result dengan perintah getResult / mendapatkan data dengan hasil data yang didapatkan berupa array. $data adalah array assosiatif yang akan dikirimkan ke dalam view, $data berisi key order dengan value $row, key pager dengan value $pager, key perPage dengan value $tampil, key total dengan value $total. Setelah itu, akan menjalankan view order/select dan juga mengirimkan $data ke dalam view.

1. Mengatur penomoran baris dari paging.

Pada view order/select :

<?php

*if* (isset($\_GET['page'])) {

        $page = $\_GET['page'];

        $jumlah = 3;

        $no = ($jumlah \* $page) - $jumlah + 1;

    }*else*{

        $no = 1;

    }

?>

Artinya, terdapat pengujian, yaitu apabila $\_GET[‘page’] di klik, maka $page akan mengambil nilai dari $\_GET[‘page’], lalu $jumlah bernilai 3, $no berisi operasi aritmatika yang hasilnya akan disimpan untuk penomoran, operasi aritmatika yaitu, $jumlah dikalikan dengan $page lalu dikurangi dengan $jumlah dan ditambah 1. Contoh $page adalah 2, maka $jumlah \* $page dikurangi $jumlah ditambah 1 adalah 3 \* 2 – 3 + 1 adalah 4, jadi nomor yang ditampilkan pada page 2, dimulai dari nomor 4. Lalu apabila $\_GET[‘page’] tidak di klik, maka nilai dari $no adalah 1.

Links dari paging :

 <?= $pager->makeLinks(1, $perPage, $total, 'bootstrap')?>

Artinya, $pager menjalankan perintah untuk makeLinks yaitu page dimulai dari page 1, lalu banyaknya data yang ditampilkan adalah sebanyak $perPage, total data yang ditampilkan mengambil nilai dari $total, dan template yang digunakan adalah menggunakan bootstrap.

1. Membuat link untuk melakukan pembayaran.

Cth :

<?php

*else* {

    $status = "<a href='".base\_url('/Admin/Order/find')."/".$value['idorder']."'>BAYAR</a>";

                        }

   ?>

Artinya, pada pengujian else, terdapat $status berisi hyperlink BAYAR yang apabila hyperlink di klik, maka akan disambungkan ke dalam base\_url() dengan alamat localhost/restoran-ci4 lalu disambungkan dengan /Admin/Order/find lalu terdapat konkret yang menyambungkan “/” dan disambung dengan $value[‘idorder’], jadi setiap BAYAR di klik, maka akan mengarahkan kepada halaman /Admin/Order/find/ lalu idorder sesuai dengan idorder pembeli.

1. Menampilkan data pada vorder dan vorderdetail sesuai dengan idorder yang dipilih.

Cth :

public function find($id = null)

    {

        $db = \Config\Database::connect();

        $sql = "SELECT \* FROM vorder WHERE idorder = $id";

        $result = $db -> query($sql);

        $row = $result -> getResult('array');

        echo "<pre>";

        print\_r($row);

        echo "</pre>";

echo "<hr>";

        $sql = "SELECT \* FROM vorderdetail WHERE idorder = $id";

        $result = $db -> query($sql);

        $detail = $result -> getResult('array');

        echo "<pre>";

        print\_r($detail);

        echo "</pre>";

    }

Artinya, pada function find mempunyai padameter $id = null, artinya nilai $id adalah kosong, lalu di dalam function find terdapat $db yang digunakan untuk menghubungkan function ke dalam database. Lalu $sql berisi perintah untuk memilih data / SELECT data dari vorder dengan memilih kolom idorder yang nilainya sama dengan $id, jadi sesuai dengan idorder yang dipilih. $result berisi $db yang menjalankan perintah query pada $sql. $row berisi $result yang berisi perintah getResult / mendapatkan data dan data hasil data akan berupa array. Lalu, $row akan ditampilkan sesuai dengan code yang ditulis dengan menggunakan print\_r, lalu hasil yang akan ditampilkan dirapikan dengan format yang lebih mudah dibaca menggunakan echo “<pre>”. Lalu hr digunakan untuk garis pemisah dengan bagian atas dan bagian bawah. Terdapat $sql kedua yang berisi perintah untuk mengambil data / SELECT data dari vorderdetail dengan memilih kolom idorder yang nilainya sama dengan $id. $result berisi $db yang menjalankan perintah query pada $sql. $detail berisi $result yang berisi perintah getResult / mendapatkan data dan data hasil data akan berupa array. Lalu, $detail akan ditampilkan sesuai dengan code yang ditulis dengan menggunakan print\_r, hasil yang akan ditampilkan dirapikan dengan format yang lebih mudah dibaca menggunakan echo “<pre>”.

1. Mengisi view order/update dengan data pelanggan sesuai dengan idorder yang dipilih.

Cth :

<div class="row mt-2">

    <div class="col"><h4><?= $judul?><h4></div>

</div>

<div class="row">

    <div class="col">

        <p>Pelanggan : <?= $order[0]['pelanggan']?></p>

    </div>

    <div class="col">

        <p>Tanggal : <?= date("d-m-yy", strtotime($order[0]['tglorder']))?></p>

    </div>

    <div class="col">

        <p>Total : <b><?= number\_format($order[0]['total'])?></b></p>

    </div>

</div>

Artinya, terapat div yang berisi $judul, lalu terdapat div row yang mempunyai 3 div col di dalamnya, col pertama terdapat tag <p> yang berisi Pelanggan dengan mengambil nilai dari $order[0][‘pelanggan’], mengambil nilai dari key order lalu karena key adalah array dan hasilnya juga berupa array, maka untuk memanggil harus dengan menggunakan [index] dengan value [kolom] yang ingin dipanggil. Col kedua di dalam tag <p> berisi tanggal dengan mengambil nilai dari $order[0][‘tglorder’], artinya mengambil data dari index 0 pada kolom tglorder, karena tglorder ini awalnya berbentuk year-month-day, maka untuk merubah format tanggal digunakan date(“day-month-year”), dan strtotime yang digunakan untuk menghasilkan waktu sesuai dengan $order[‘tglorder’] dengan date nya adalah d-m-yy. col ketiga berisi tag <p> dengan Total yang mengambil nilai dari $order[0][total], artinya mengambil nilai dari index 0 dan kolom total.

1. strtotime() adalah string to time, berfungsi untuk menghasilkan waktu tertentu dengan format timestamp.
2. Melakukan update data pada kolom bayar setelah selesai melakukan pembayaran.

Cth :

public function update()

    {

        $idorder = $\_POST['idorder'];

        $total = $\_POST['total'];

        $bayar = $\_POST['bayar'];

*if* ($bayar < $total) {

            session() -> setFlashdata('info', 'Mohon maaf, pembayaran anda kurang!');

            $this -> find($idorder);

        } *else* {

            $db = \Config\Database::connect();

            $kembali = $bayar - $total;

            $sql = "UPDATE tblorder SET bayar = $bayar, kembali = $kembali, status = 1 WHERE idorder = $idorder";

            $db -> query($sql);

*return* redirect() -> to(base\_url("/Admin/Order"));

        }

    }

Artinya, karena pada views order/update terdapat form dengan name bayar, idorder, dan total, form tersebut mempunyai action yang tindakan nya diarahkan ke dalam /Admin/Order/update, lalu di dalam function update terdapat $idorder yang mengambil nilai dari $\_POST[‘idorder’], $total yang mengambil nilai dari $\_POST[‘total’], dan $bayar yang mengambil nilai dari $\_POST[‘bayar’]. Lalu terdapat pengujian, apabila $bayar nilai nya lebih besar dari $total, maka akan menjalankan session() yang berisi perintah untuk setFlashdata dengan key info dan pesan yang ditampilkan adalah Mohon maaf, pembayaran anda kurang, lalu akan diarahkan kembali ke function find dan menjalankan $idorder sesuai dengan $\_POST[‘idorder’]. Apabila nilai $bayar lebih besar dari $total, maka akan menjalankan $db yang digunakan untuk menyambungkan function ke dalam database, lalu $kembali yang menyimpan hasil dari operasi aritmatika dari $bayar - $total, $sql berisi perintah untuk mengubah data / UPDATE data dengan mengupdate kolom bayar sesuai dengan $bayar, kolom kembali sesuai dengan $kembali, dan status diganti menjadi 1, kolom yang diubah sesuai dari kolom $idorder yang dipilih. Lalu $db menjalankan perintah query pada $sql, setelah berhasil negupdate data pada kolom – kolom setelah melakukan pembayaran, maka halaman akan dikembalikan ke halaman base\_url() dengan alamat localhost/restoran-ci4 lalu disambungkan dengan /Admin/Order.

1. Menampilkan pesan menggunakan getFlashdata.

Cth :

<div class="row">

    <div class="col mt-1">

        <?php

*if* (!empty(session() -> getFlashdata('info'))) {

                echo '<div class="alert alert-danger" role="alert">';

                echo session() -> getFlashdata('info');

                echo '</div>';

            }

        ?>

    </div>

</div>

Artinya, di dalam div row terdapat pengujian yaitu, apabila session() –> getFlashdata dengan key info tidak kosong, maka akan menampilkan session() yang menjalankan perintah getFlashdata / perintah untuk mengambil data pada key info, lalu pesan diberi background merah.

1. Melakukan pencarian tanggal yang mengambil data dari database.

Cth :

    public function cari()

    {

*if* (isset($\_POST['awal'])) {

            $db = \Config\Database::connect();

            $awal = $\_POST['awal'];

            $sampai = $\_POST['sampai'];

            $sql = "SELECT \* FROM vorderdetail WHERE tglorder BETWEEN '$awal' AND '$sampai' ";

            $result = $db -> query($sql);

            $row = $result -> getResult('array');

            $data = [

                'judul' => 'DETAIL ORDER',

                'orderdetail' =>$row

            ];

            echo view("orderdetail/cari", $data);

        }

    }

Artinya, di dalam function cari terdapat pengujian, apabila $\_POST[‘awal’] di klik, maka akan menjalankan $db yang digunakan untuk menyambungkan ke dalam database, lalu $awal mengambil nilai dari $\_POST[‘awal’] dan $sampai mengambil nilai dari $\_POST[‘sampai’], lalu $sql berisi perintah untuk memilih data / SELECT data vorderdetail dengan kolom yang dipilih adalah tglorder dan memilih data dari tglorder diantara $awal dan $sampai. $result berisi $db yang menjalankan perintah query pada $sql, $row berisi $result yang menjalankan perintah untuk getResult / mengambil data dan hasil data yang akan ditampilkan akan berupa array. $data beris data – data yang akan dikirimkan ke dalam view pada orderdetail/cari, $data berisi key judul dengan value DETAIL ORDER, key orderdetail dengan value $row. Lalu menjalankan view orderdetail/cari dan menjalankan $data.

1. Menambahkan hyperlink ke dalam $aktif yang jika link di klik, maka akan berubah status nya, dari aktif menjadi tidak aktif atau dari tidak aktif menjadi aktif.

Cth :

Di dalam file views user/select :

<td>

<a href=

"<?= base\_url('/Admin/User/update'."/".$value['iduser']."/".$value['aktif'])?>">

<?= $aktif?>

</a>

</td>

Artinya, terdapat hyperlink pada $aktif yang apabila $aktif di klik, maka akan menjalankan base\_url() dengan alamat localhost/restoran-ci4 lalu disambungkan dengan /Admin/User/update/$value[‘iduser’]/$value[‘aktif’], sehingga hyperlink akan menjalankan function update di dalam user dan membaca value dari iduser dan value dari aktif.

Di dalam function update :

public function update($id = null, $isi = 1)

    {

        $model = new User\_M();

*if* ($isi == 1) {

            $isi = 0;

        } *else* {

            $isi =1;

        }

        $data = [

            'aktif' => $isi

        ];

        $model -> update($id, $data);

*return* redirect() -> to(base\_url("Admin/User"));

    }

Artinya, karena apabila hyperlink pada $aktif di klik, maka halaman akan diarahkan ke dalam halaman function update, di dalam function update, terdapat parameter yang berisi $id bernilai null, artinya $id mempunyai nilai kosong, dan $isi yang berisi 1. Lalu menjalankan objek $model yang berisi User\_M, dimana User\_M ini berisi $table tbluser, artinya tabel yang disambungkan oleh model ke dalam database adalah tbluser, dan terdapat $primaryKey ‘iduser’, artinya primary key dari tbluser adalah iduser, primary key digunakan untuk melakukan delete data dan update data. Setelah menjalankan $model, terdapat pengujian dengan if, yaitu apabila $isi bernilai 1, maka $isi akan bernilai 0, dan jika nilai dari $isi adalah 0, maka $isi akan mempunyai nilai 1. Setelah itu, $data berisi data yang akan dirubah data / update, yaitu key aktif dengan value $isi. $model berisi perintah untuk update data sesuai dengan $id dan $data adalah data yang diupdate. Setelah berhasil menjalankan update data, maka halaman akan dikembalikan ke halaman base\_url() lalu diarahkan ke /Admin/User.

1. Melakukan validasi pada insert user.

Cth :

Di dalam model User\_M :

class User\_M extends Model

{

    protected $table = 'tbluser';

    protected $allowedFields = ['user', 'email', 'password', 'level', 'aktif'];

    protected $validationRules = [

        'user' => 'alpha\_numeric\_space|min\_length[3]|is\_unique[tbluser.user]',

        'email' => 'valid\_email'

    ];

    protected $validationMessages = [

        'user' =>[

            'alpha\_numeric\_space' => 'Mohon maaf, di dalam pengisian user, tidak boleh menggunakan simbol,

            harus diisi dengan huruf, angka, dan spasi.',

            'min\_length' => 'Isi minimal dari data user yang dimasukkan adalah 3 huruf',

            'is\_unique' => 'Nama user sudah terdaftar, silahkan menggunakan nama yang lain'

        ],

        'email' =>[

            'valid\_email' => 'Silahkan masukkan email dengan valid'

        ]

    ];

}

Artinya, di dalam User\_M terdapat $table tbluser, berarti tabel yang dihubungkan oleh model ke dalam database adalah tbluser, $allowedFields berisi kolom kolom yang harus diisi apabila akan melakukan insert data, yaitu kolom user, email, password, level dan aktif, $validationrules berisi peraturan peraturan pada user dan email, dan $validationmessages berisi pesan pesan yang akan ditampilkan ketika tidak sesuai dengan peraturan.

public function insert()

    {

*if* (isset($\_POST['simpan'])) {

            $data = [

                'user' => $\_POST['user'],

                'email' => $\_POST['email'],

                'password' => password\_hash($\_POST['password'], PASSWORD\_DEFAULT),

                'level' => $\_POST['level'],

                'aktif' => 1

            ];

            $model = new User\_M();

*if* ($model -> insert($data)===false) {

                $error = $model->errors();

                session() -> setFlashdata('info', $error);

*return* redirect()->to(base\_url("/Admin/User/create"));

            } *else* {

*return* redirect()->to(base\_url("/Admin/User"));

            }

        }

    }

Artinya, di dalam function insert terdapat pengujian, yaitu, apabia $\_POST[‘simpan’] di klik, maka akan menjalankan, $data yang berisi data – data yang datanya akan di INSERT kan ke dalam database, nama key dari $data harus sama dengan nama kolom yang ada di tabel yang akan dimasukkan datanya, yaitu tbluser, $data berisi, key user mengambil nilai dari $\_POST[’user’], key email mengambil nilai dari $\_POST[‘email’], key password mengambil nilai dari $\_POST[‘password’] yang diletakkan di dalam password\_hash, yaitu perintah untuk mengeknkripsi password menjadi tidak diketahui oleh orang lain, lalu key level mengambil nilai dari $\_POST[‘level’], dan key aktif berisi nilai 1. Setelah mengambil data yang akan di insertkan, lalu memanggil model yang akan dijalankan, yaitu $model yang berisi model User\_M. terdapat pengujian kedua, yaitu untuk mengecek data yang dimasukkan, apabila data yang dimasukkan / $model yang menjalankan perintah insert pada $data bernilai salah, di dalam $error terdapat $model yang berisi perintah errors. Setelah itu, menjalankan session() yang berisi perintah untuk mengatur flashdata / setFlashdata dengan key info dan menjalankan $error. Setelah selesai mengatur flashdata, halaman akan dikembalikan ke /Admin/User/create, pesan error akan ditampilkan menggunkaan perintah getFlashdata yang ada di dalam file user/insert .

Tetapi, apabila data yang dimasukkan sesuai dengan peraturan, maka data berhasil dimasukkan / berhasil melakukan INSERT. Setelah itu akan dikembalikan ke dalam halaman /Admin/User.

1. Mengambil nilai dari $\_POST dengan menggunakan request agar data yang diambil ada keamanannya karena halaman nya diakses oleh banyak orang, jadi apabila ingin masuk ke dalam web, maka harus login terlebih dahulu.

Cth :

Di dalam function index :

public function index()

    {

*if* ($this -> request -> getMethod() == 'post') {

            $email = $this -> request -> getPost('email');

            $password = $this -> request -> getPost('password');

            $model = new User\_M();

            $user = $model -> where(['email'=>$email, 'password' => $password, 'aktif' => 1]) -> first();

            echo "<pre>";

            print\_r($user);

            echo "</pre>";

            $this -> setSession($user);

        }

Artinya, karena form yang dikirimkan mempunyai action /Admin/Login, maka secara otomatis akan mengarahkan ke dalam halaman /Admin/Login, lalu nilai dari $\_POST yang dikirimkan otomatis akan tersimpan ke dalam function index pada /Admin/Login, sehingga di dalam function index terdapat pengujian, apabila request yang menjalankan perintah getMethod mempunyai nilai post, maka akan menjalankan $email, yang mengambil nilai dari request yang menjalankan perintah untuk mengambil nilai dari post / getPost [‘email’], lalu $password yang mengambil nilai dari request yang berisi perintah untuk mengambil nilai dari post / getPost [‘password’]. Pengambilan nilai dari $\_POST menggunakan request agar data terbungkus. Setelah itu, menjalankan objek $model berisi model User\_M, yang berisi $table tbluser, artinya tabel yang dihubungkan oleh model ke dalam database adalah tbluser. Lalu $user berisi $model dengan menjalankan perintah where, nama dari key di dalam where, sesuai dengan nama kolom yang dipilih pada tbluser, untuk mengambil data dari key email dengan value sesuai dengan $email, lalu key password dengan value sesuai dengan $password, dan key aktif dengan nilai 1, setelah melakukan seleksi menggunakan where, terdapat fisrt yang digunakan untuk mengambil data pertama yang cocok dari seleksi tersebut. Setelah menjalankan $user, data ditampilkan meggunakan print\_r agar data yang ditampilkan sesuai dengan yang ditulis, dan untuk merapikan format tampilan menggunakan echo “<pre>”. Lalu, membaca function setSession yang menjalankan $user.

Di dalam function setSession :

    public function setSession($user)

    {

        $data = [

            'user' => $user['user'],

            'email' => $user['email'],

            'level' => $user['level'],

            'loggedin' => true

        ];

        session() -> set($data);

    }

Artinya, di dalam function setSession dengan parameter yang membaca $user, lalu di dalam function setSession berisi $data yang akan diatur isinya ke dalam session, $data berisi key user dengan value mengambil nilai dari $user[‘user’], key email dengan value mengambil nilai dari $user[‘email’], key level mengambil nilai dari $user[‘level’], dan key loggedin berisi nilai true. Setelah itu, di dalam session() diatur isi dari session adalah $data.

Session() ditampilkan di dalam admin.php :

<div>

          <?php

*if* (!empty(session() -> get('user'))) {

              echo session() -> get('user');

                   }

            ?>

                    </div>

Artinya, terdapat pengujian yaitu apabila session() -> get(‘user’) tidak kosong, maka akan menampilkan session() yang berisi perintah untuk get / unutk mendapatkan data dari key user, sesuai dengan session() yang di set.

Pemahaman untuk saya sendiri agar ingat, jadi setelah form login diisi lalu menjalankan function index, di dalam function index mengambil data user sesuai dengan email dan password yang dimasukkan, dan sesuai dengan status aktif = 1, lalu membuat function setSession yang membaca $user digunakan untuk menyimpan data, di dalam setSession, terdapat $data berisi nilai yang diambil dari $user dengan membaca kolom, user, email, dan level. Setelah itu, session() berisi perintah untuk mengatur $data atau set $data. Di dalam function index, terdapat $this -> setSession($user), yang digunakan untuk menjalankan setSession yang di dalamnya berisi session yang sudah diatur, jadi setSession akan membaca nilai dari email dan password yang dimasukkan, lalu akan mengambil data user dengan nilai email dan password yang sesuai, lalu mengambil key di dalam $data dan menyimpannya. Setelah itu, untuk menampilkan session(), di dalam admin.php menggunakan session() -> get(‘user’), sehingga data diambil dengan perintah get dari key user yang sudah diatur pada function setSession.

1. First() digunakan untuk mengambil data yang paling atas setelah melakukan seleksi dan data tersebut sesuai dengan apa yang diseleksi. Jadi ketika menyeleksi data, tetapi ada 2 atau 3 data yang sama, maka yang diambil adalah data pertama yang cocok.
2. data yang dikirimkan menggunakan array assosiatif bisa dituliskan dengan 2 cara.

Cth :

Cara pertama :

 $data['info'] = "User atau Password salah";

Cara kedua :

$data = [

                'info' => "User atau Password salah"

            ];

Cara pertama dan cara kedua bernilai sama, dan jika ingin melakukan pemanggilan, maka yang dipanggil adalah key nya, seperti $info. Cara pertama digunakan apabila data yang dimasukkan adalah data tunggal dan cara kedua digunakan apabila data yang dimasukkan tunggal atau banyak.

1. Filter pada codeigniter digunakan untuk mengecek controller mana yang akan di filter.
2. Untuk menggunakan filter, hal yang harus dilakukan pertama kali adalah dengan membuat file Auth.php di dalam folder Filters, di dalam Auth.php hanya berisi function. Di dalam function before terdapat pengujian yang akan diarahkan ke dalam /Admin/restoran-ci/Login. Setelah itu, membuat alias pada file Filters.php digunakan untuk menyalakan / menyambungkan file pada folder filters yang telah dibuat. Terakhir, untuk mengatur jalannya routes, maka di dalam Routes.php diberi [‘filter’ => ‘Auth’] untuk menjalankan filter Auth yang sudah dibuat, karena di dalam Auth.php akan diarahkan ke dalam /Admin/Login, maka semua $routes yang dipasang [‘filter’ => ‘Auth’] harus selesai masuk ke dalam /Admin/Login setelah itu baru menjalankan isi dari $routes. .

Isi dari Auth.php :

    public function before(RequestInterface $request, $arguments = NULL)

    {

*if* (!session() -> get('loggedIn')) {

*return* redirect()->to(base\_url('/Login'));

        }

    }

Terdapat pengujian yaitu apabila session() -> get(‘loggedIn’) tidak terisi, karena (‘loggedIn’) bernilai true, maka jika loggedIn tidak bernilai true , sehingga halaman akan diarahkan ke dalam basae\_ur()/Login.

Alias dari Auth agar dapat tersambung :

public $aliases = [

        'Auth' => \App\Filters\Auth::class,

    ];

Key Auth harus sama dengan nama dari file di dalam folder Filters yang dibuat, dan value nya adalah di dalam App lalu masuk ke dalam Folder Filters lalu ke dalam file Auth dan menjalankan class.

Jadi, semua routes yang diberi [‘filter’ => ‘Auth’] akan dijalankan ketika selesai melakukan Login.

1. password\_verify() digunakan untuk mencocokkan passwordyang sudah dienkripsi pada data yang dimasukkan dan password yang ada di dalam database.
2. Melakukan enkripsi password di dalam function index.

Cth :

public function index()

    {

        $data = [];

*if* ($this -> request -> getMethod() == 'post') {

            $email = $this -> request -> getPost('email');

            $password = $this -> request -> getPost('password');

            $model = new User\_M();

            $user = $model -> where(['email'=>$email, 'aktif' => 1]) -> first();

*if* (empty($user)) {

                $data['info'] = "Email yang anda masukkan salah!";

            } *else* {

*if* (password\_verify($password, $user['password'] )) {

                    $this -> setSession($user);

*return* redirect() -> to(base\_url("/Admin"));

                } *else* {

                    $data['info'] = "Password yang anda masukkan salah!";

                }

            }

        }

        echo view("template/login", $data);

    }

Artinya, karena form yang dikirimkan mempunyai action /Admin/Login, maka secara otomatis akan mengarahkan ke dalam halaman /Admin/Login, lalu nilai dari $\_POST yang dikirimkan otomatis akan tersimpan ke dalam function index pada /Admin/Login, sehingga di dalam function index terdapat pengujian, apabila request yang menjalankan perintah getMethod mempunyai nilai post, maka akan menjalankan $email, yang mengambil nilai dari request yang menjalankan perintah untuk mengambil nilai dari post / getPost [‘email’]. Pengambilan nilai dari $\_POST menggunakan request agar data terbungkus. Setelah itu, menjalankan objek $model berisi model User\_M, yang berisi $table tbluser, artinya tabel yang dihubungkan oleh model ke dalam database adalah tbluser. Lalu $user berisi $model dengan menjalankan perintah where, nama dari key di dalam where, sesuai dengan nama kolom yang dipilih pada tbluser, untuk mengambil data dari key email dengan value sesuai dengan $email, lalu key password dengan value sesuai dengan $password, dan key aktif dengan nilai 1, setelah melakukan seleksi menggunakan where, terdapat fisrt yang digunakan untuk mengambil data pertama yang cocok dari seleksi tersebut. Setelah melakukan seleksi data, terdapat pengujian, yaitu apabila $user kosong, maka akan menjalankan $data dengan key info yang mempunyai value Email yang anda masukkan salah!, lalu apabila $user terisi, maka akan menjalankan perintah untuk mencocokkan password dari $password yaitu dari password yang dimasukkan dan mencocokkannya dengan $user[‘password’], setelah selesai dicocokkan, apabila password cocok, maka akan menjalankan function setSession yang mengambil nilai dari $user, lalu halaman akan dikembalikan ke dalam halaman base\_url() lalu disambungkan dengan /Admin. Tetapi apabila password yang dimasukkan tidak cocok, maka akan menjalankan key info dengan tulisan “Password yang anda masukkan salah”. Setelah itu akan menjalankan template/login dan menjalankan $data apabila terjadi kesalahan dalam memasukkan email atau apabila password tidak cocok.

**Saya Belum Mengerti**